MINI PROJECT REPORT

Submitted to the faculty of Engineering and Technology

VI Semester, B.Tech

(Autonomous Batch)

IMPLEMENTATION OF VARIOUS FILTERING TECHNIQUES

BY
BADAM JWALA SRI HARI
B18CS003

Under the Guidance of
P. Niranjan Reddy
Professor

Department of Computer Science & Engineering

Kakatiya Institute of Technology & Science

(An Autonomous Institute under Kakatiya University)

Warangal (Telangana State)

2020-21

KAKATIYA INSTITUTE OF TECHNOLOGY & SCIENCE

Warangal – 506 015, Telangana, INDIA. *(An AUTONOMOUS INSTITUTE under Kakatiya University, Warangal)*

కాకతీయ సాంకేతిక విజ్ఞాన శాస్త్ర విద్యాలయం, వరంగల్ – ౫౦౬౦౧౫.

Estd-1980

KITSW

CERTIFICATE

This is to certify that **BADAM JWALA SRI HARI** bearing roll no: **B18CS003** of the VI Semester B.Tech. Computer Science and Engineering (Autonomous) has satisfactorily completed the Mini Project Report entitled **"Implementation of various filtering techniques".**

Supervisor

P. Niranjan Reddy

Professor

Cordinator	**Convener**
C. Madan Kumar	Dr. Kumar Dorthi
Asst. Professor	Asst. Professor

Head of the Department

Prof. V. Shankar

ACKNOWLEDGMENT

I extend my sincere and heartfelt thanks to our esteemed guide, **P.Niranjan Reddy, Professor** for his exemplary guidance, monitoring and constant encouragement throughout the course at crucial junctures and for showing us the right way.

I am grateful to respected coordinator, **C. Madan Kumar, Asst. Professor** for guiding and permitting me to utilize all the necessary facilities of the Institute.

I sincere thanks to respected convener **Dr. Kumar Dorthi, Asst. Professor** for supporting me and to utilize all the necessary facilities of the Institute.

I would like to extend thanks to our respected head of the department, **Prof. V. Shankar** for allowing us to use the facilities available. I would like to thank other faculty members also.

I express my sincere thanks to **Dr. K. Ashoka Reddy, Principal**, KITS, Warangal, for his kind gesture and support.

I would like to thank all the faculty members, friends and family for the support and encouragement that they have given us during the Mini project.

BADAM JWALA SRI HARI

B18CS003

ABSTRACT

Now a days many things going around the image processing because there are lot application based on image. In image processing after taking the image as input the first and foremost step is pre-processing. The main objectives of pre-processing is to remove noises from the image. In the perceptive of medical field converting the image in gray image also adds in pre-processing step. Removal of noise in the image is done by using filtering techniques. Filtering techniques changes the image based on shape, size, color, depth, etc. There are varies types of noises like salt pepper noises, Blurness ,Gaussian noises, periodic noises, etc .This noises are removed by varies filtering like mean filters , Median filters, Weiner filters and many smoothing techniques. This project is to implement filtering techniques on a noisy image.

ACRONYMS

FFT	Fast Fourier Transform
LP	Low Pass
HP	High Pass
ILP	Ideal Low Pass
IHP	Ideal High Pass
BWLP	Butterworth Low Pass
BWHP	Butterworth High Pass
GLP	Gaussian Low Pass
GHP	Gaussian High Pass
CV	Computer Vision

LIST OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Introduction

Image processing is a method of performing some operations on an image, in order to get an enhanced image or to extract some useful information from it. Image processing takes an image as input and output may be image or characteristics/features associated with that image.

Basic steps in image processing are

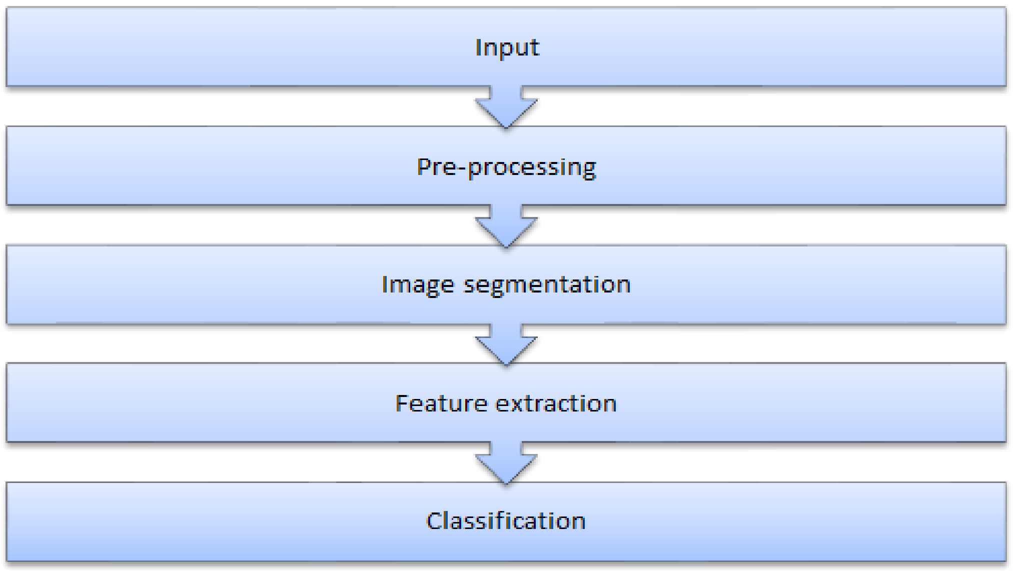


Fig 1: steps of image processing

In first step input image is acquisition will take place. Means image is captured from sensors and that captured continuous image is converted into digitized image with the help of sampling and quantization. Second step is pre-processing .In pre-processing noise is removed from an image by using histogram processing, filter techniques, morphological operators. Image is converted into gray scale image and resize are also takes place based on the requirement.

In third step partition of output image from pre-processing will take place. The main goal of image segmentation is to change the image representation in such a way that it should be easy to analyzer in further step. Fourth step is feature extraction. Feature extraction refers to both detecting the features and describing the detected feature. Last step of image processing is to classify the image based on the features extraction in the above step.

1.2. Objective

The main objective of this project is

Filters are subjective. When a user is building an image processing model then the first and fore most step after image acquisition is pre-processing filters are applied in order to remove noises use have to change filters from model to model based on the requirement. Main aim to build a model such that the user who is developing the model should know which filter to apply in order to decrease the noises

CHAPTER 2

LITERATURE REVIEW

There are many researches made on filters and denoising the image because now a days the role of image processing is increasing in every sector

In [1] they discussed about how filters are classified into based on the property of linearity, the algorithm of that filters are explained. They clearly explained difference between Gaussian filter and bilateral filter. Ongoing through this can know about the applications of image processing in various sectors.

In [2] the idea of author is combine more than one sharpening filters in order to increases the look and feel of the image. Discussed about first order derivative, second order derivative and how they used in Sobel, Laplacain.

In [3] they consider two filter from spatial domain mean and Weiner filter. They did experimentation on medical image and they considered a three noises they are speckle noise, salt and pepper noise and Gaussian.

In [4] the main concern is about frequency domain. How frequency domain is classified into varies type .discuss about low and high pass filter and they briefly mentioned about what are the operation we can perform on the image like Image enhancement, Image analysis, image restoration, image compression, image synthesis.

In [5] classification of filters is explained in that paper. Image denoise methods are classifed into spatial domain and Transform domain. Explained about how Transform domain are futherly divided into

CHAPTER 3

IMPLEMENTATION

3.1. Algorithm:

Filters are used in pre-processing to remove noise from an image and enhancing the look and feel of an image. Filtering is the process of removing noises from the image. Enhancement is the process of manipulating an image so that the result is more suitable than the original for a specific application.

Image enhancement filters are basically classified into two types

i. Spatial domain filters
ii. Frequency domain filters

In spatial domain we operate directly on the pixels of an image. In frequency domain operation are performed on the fourier transform of an image rather than on the image itself.

Each domain further classified into two types

a. Smoothening filters
b. Sharpening filters

Smoothening is the process of applying blurness to the image in order to reduces noises unlike to smoothening filters sharpening filters are used to highlight the edges in an image. Examples of Smoothing filters are Mean, Gaussian, Median, Min, Max, etc and Examples for Sharpening filters are Laplacian , Roberts cross gradient , Sobel, etc.

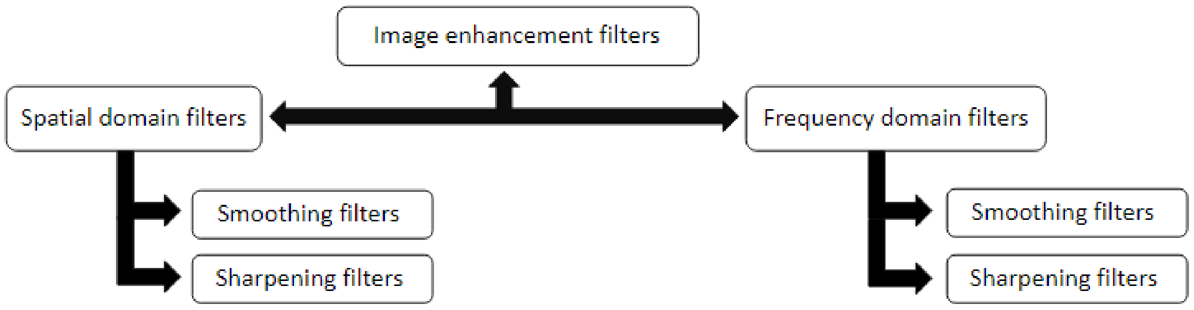


Fig 2: basic classification of Image enhancement filters

3.1.1 Spatial Domain filters:

In Spatial domain filtering operations on the neighborhood of every pixel of an image.

Basic terminology:

- Kernel: Kernel is the matrix which moves on the image and performs operations based on the filter. This kernel is also referred as mask/window.
- Correlation: The process of applying kernel on an image.
- Convolution: Rotating kernel by 180^0 and applying.
- If kernel is symmetric about center then correlation and convolution produces same results

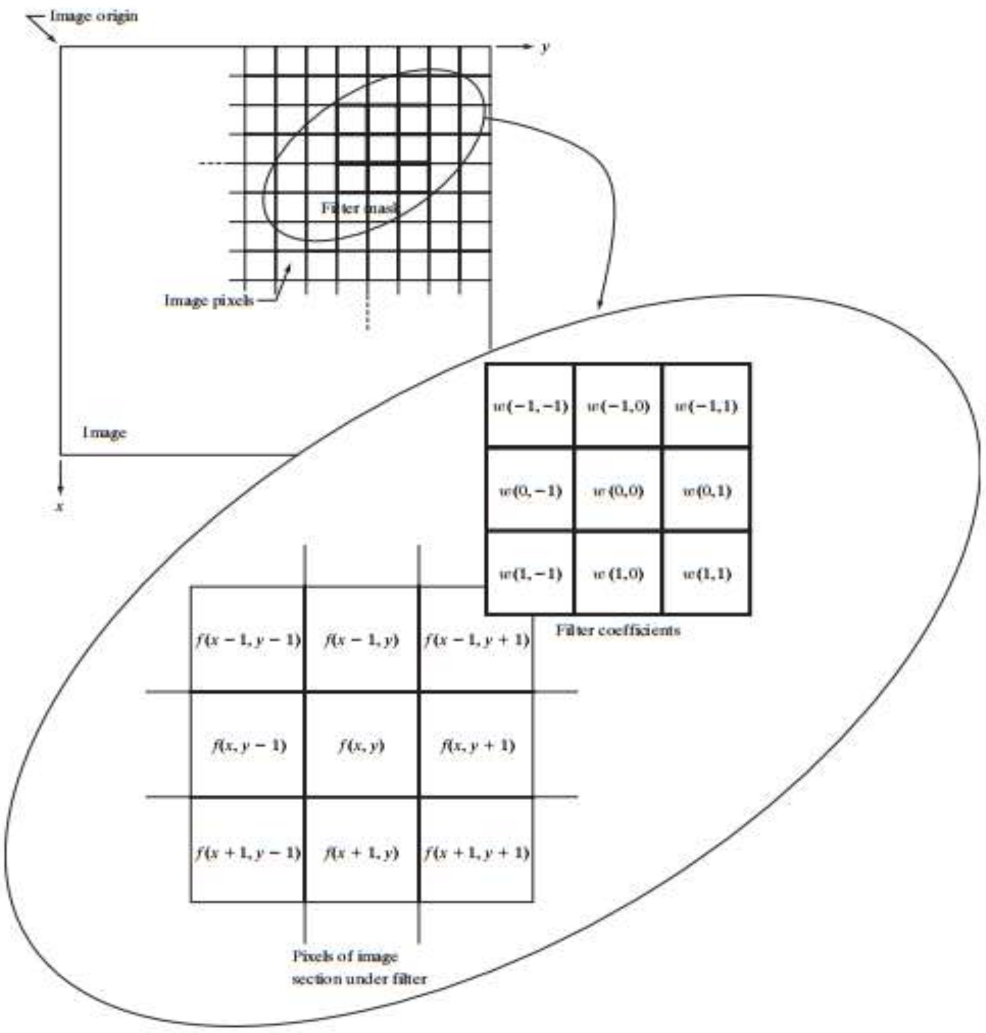


Fig 3: showing that how kernel is sliding on image

Every filter in the spatial domain can be either linear or non-linear filter.

- Linear filter: sum of product operation is performed between kernel and image in linear spatial filter.
- Non Linear filter: Non-linear filter ranks the pixels covered by the kernel on the image. Based on the filter type it replaces the center pixel.

Based on the linearity property smoothing and sharpening filters are divided as below

Linear filters	**Non- Linear filters**
■ Mean ---------- Smoothing	■ Median --------- Smoothing
■ Gaussain -------- Smoothing	■ Min ------------- Smoothing
■ Laplacian ------- Sharpening	■ Max ------------- Smoothing
■ Roberts --------- Sharpening	
■ sobels ---------- Smoothing	

Linear filters:

Mean:

Mean filter is used to smoothing an image by applying the blurness. Mean filter takes the sum of pixels which are encompassed by the kernel and sum is normalized by dividing with total no of pixels covered by the filter. Mean filter is also known as Average filter or Box filter.

$$R = \frac{1}{9}\sum_{i=1}^{9} z_i$$

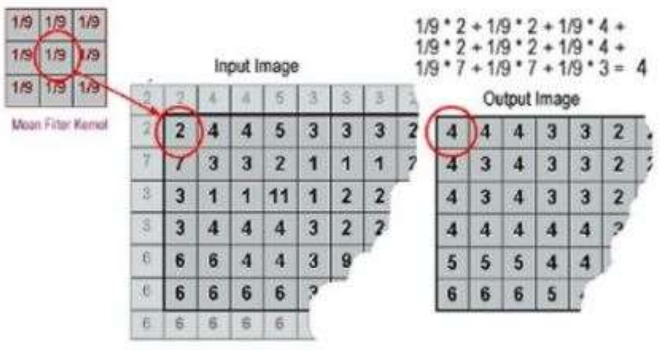


Fig 4: Mean filter

Gaussian filter:

Gaussian filter is also called as weighted average filter which is similar to mean filter the only different is kernel/mask. The Gaussian filter is generated on bases of Normal distribution where mean is Zero. As the image is represented in 2D so we have to consider 2D normal distribution equation. The graph drawn based on the 2D Normal distribution equation is shown below. The intensity is highest at the center and gradually decreases when it moves to edges and corners as shown in the graph and we can simulate the same with filter of size 21 X 21 in the below figure.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \qquad G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

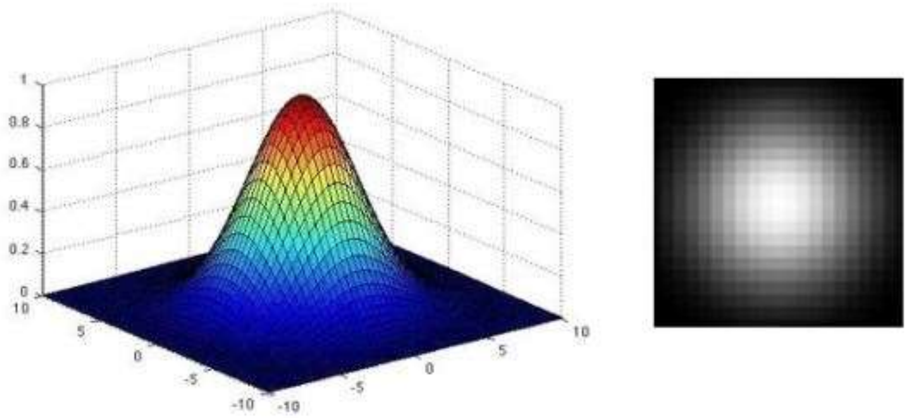


Fig 5: Gaussian filter

Laplacian filter:

Laplacian filter is an image sharpening filter. Means this filter highlights the edges based on the intensity. In order to construct a kernel we have to formulate a second order derivative equation from that equation we use coefSficients to build a filter. We had a laplacian operator with two variables x, y. Only two because as we can represented an image by 2 D. laplacian filter is isometric means it is rotation invariant. It produces grayish outline and other discontinuous.

Laplacian operator is

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

-------------------- equation 1

First order derivative of ‘f’ with respected to ‘x’ is

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

On differentiating the above equation with respected to ‘x’ gives

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

--------------- equation 2

Similarly in ‘ y ‘ direction

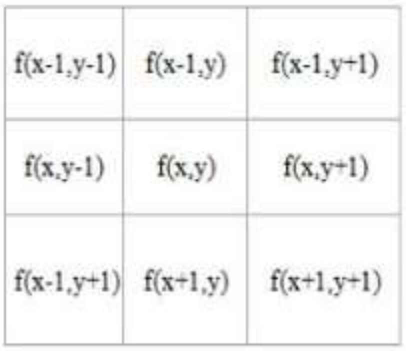
$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

--------------- equation 3

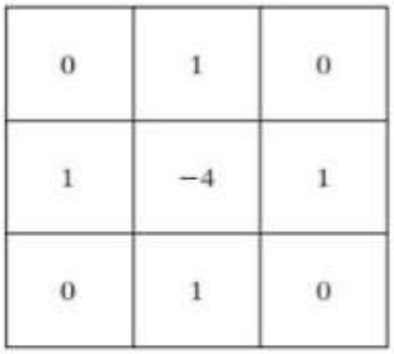
On Substitute equation 2 and equation 3 in equation 1 then we get

$$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) -4f(x, y)$$

On substituting the coefficients in respected positions we will get the kernel.



f(x-1,y-1)	f(x-1,y)	f(x-1,y+1)
f(x,y-1)	f(x,y)	f(x,y+1)
f(x-1,y+1)	f(x+1,y)	f(x+1,y+1)



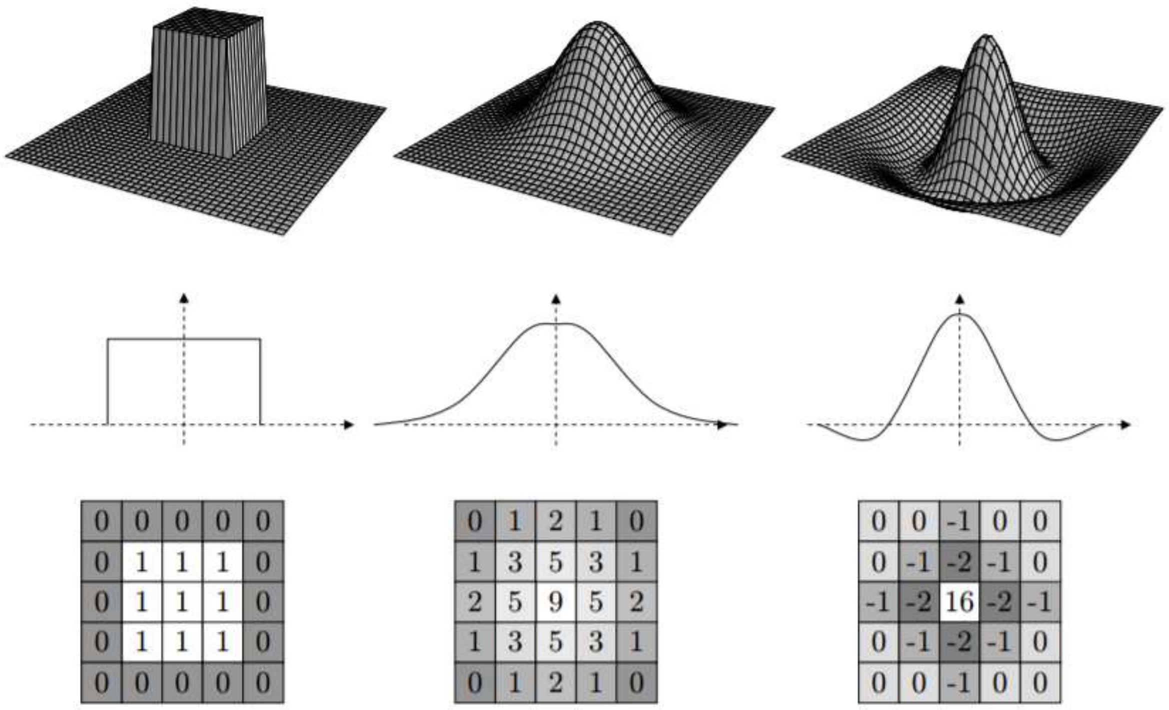
0	1	0
1	−4	1
0	1	0

Fig 6: kernel of laplacian filter

Differences between Mean, Gaussian, Laplacian filter:

1st column is about mean filter and 2nd, 3rd are about Gaussian and Laplacian filters respectively.

From below image we can clearly observe how graphs and kernels are related. In mean filter graph values are constant. So, the values in the kernel are same (one). In Gaussian filter Intensity values are high at center and gradually decrease when we are moving to edges and corners. In Laplacian graph there are some values below the x- axis and we can observe there are negative in the kernel



0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Fig 7: difference between mean, median, laplacian

Roberts filter:

Roberts filter is also used for sharpening an image. In Roberts kernel is constructed based on first order derivative. In image processing first order derivation is implemented using magnitude of gradient.

$$\nabla f \equiv \operatorname{grad}(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

Partial derivatives are not isotropic but magnitude of gradient is isotropic

$$M(x, y) = \operatorname{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

Still the about expression can preserves relative changes but cannot preserves isotropic property. The smallest approximations from first order derivative is satisfied by $g_x = (z_9 - z_5)$ and $g_y = (z_8 - z_5)$. Coefficients of z's are replaced in that 3 X 3 matrix.

In g_x coefficients of $z_9 = 1$ and $z_5 = -1$ and

In g_y coefficients of $z_8 = 1$ and $z_5 = -1$.

On substituting g_x and g_y in about results to

$$M(x, y) = \left[(z_9 - z_5)^2 + (z_8 - z_6)^2\right]^{1/2}$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0
0	1

0	−1
1	0

Fig 8 (a) 3 X 3 z's as intensities (b) and (c) are Roberts cross gradient

Sobel filter:

Sobel filter is used for sharpening. As the Roberts filter is not symmetric about center. So, that disadvantages is over came by sobel filter. Means here Approximations values satisfied by g_x and g_y should results in 3 x 3 as filter should be symmetric about center. the approximations satisfied by g_x and g_y are

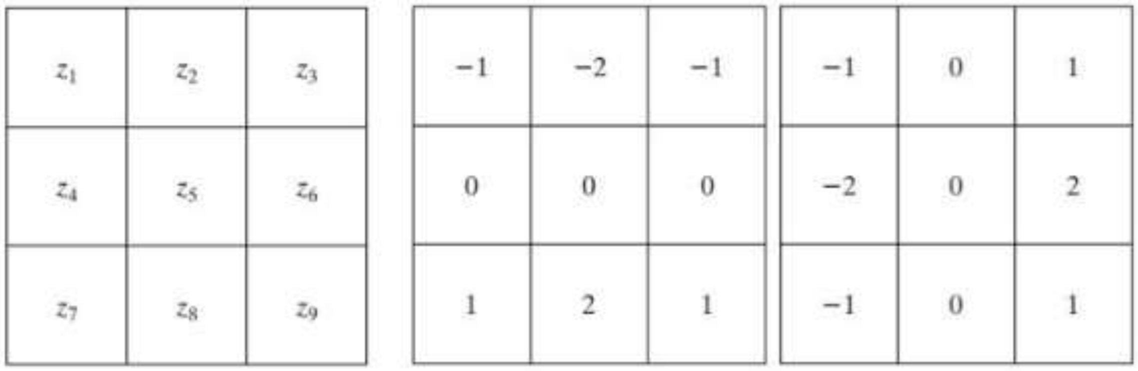
$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

Coefficients of z's are replaced in that 3 X 3 matrix to get kernels.

In g_x coefficients of $z_7 = 1$, $z_8 = 2$, $z_9 = 1$, $z_1 = - 1$, $z_2 = -2$, $z_3 = -1$ and

In g_y coefficients of $z_3 = 1$, $z_6 = 2$, $z_9 = 1$, $z_1 = -1$, $z_4 = -2$, $z_7 = -1$.



z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Fig 9: (a)3 X 3 z's as intensities (b) and (c) sobel operators g_x and g_y

Non –linear filters:

Median filters:

Median filter is a smoothing filter. A window slides through the images. The values covered by the window should by ranked and find the median in that covered pixels and replace the center pixel covered by kernel with median value.

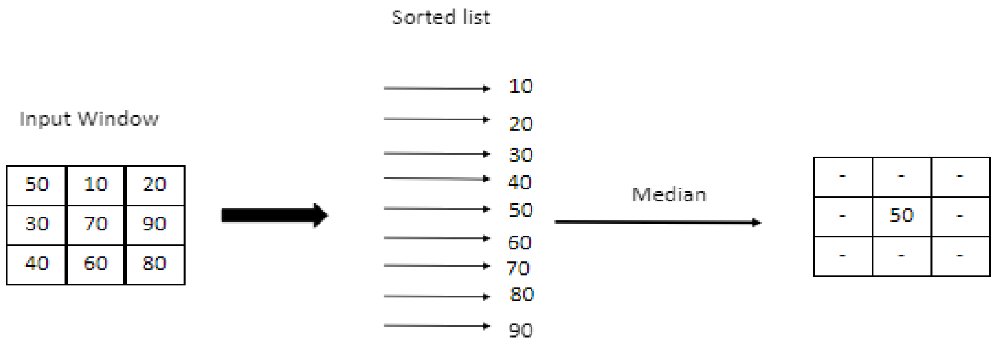


Fig 10: Median filter

Min and max filters:

Min and Max filters are also smoothing. Min and Max filters are similar to median filter. The values covered by the window should be ranked and find the minimum and maximum and replace the center pixel covered by kernel with min and max values respectively.

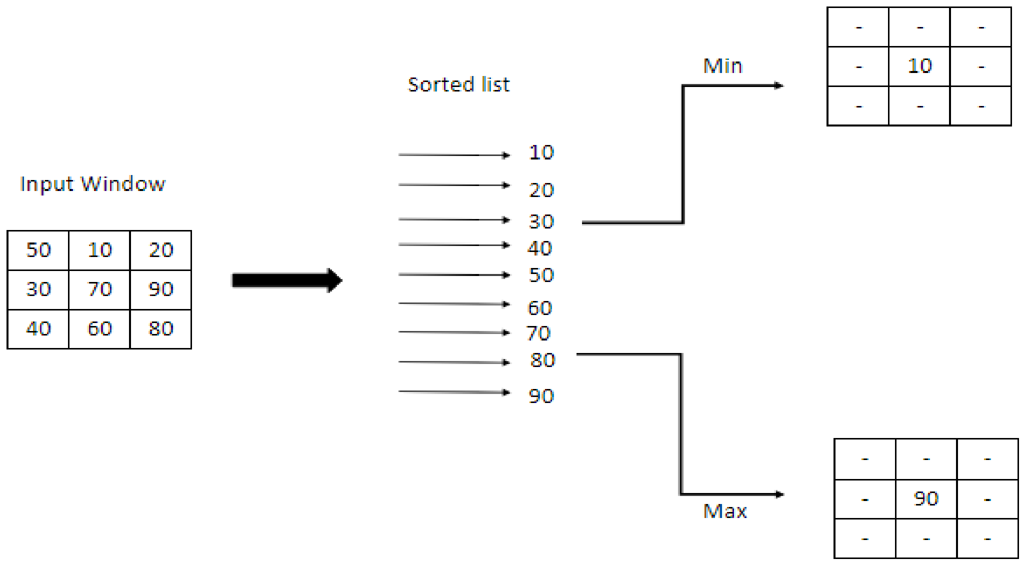


Fig 11: min and max filters

3.1.2 Frequency Domain filters

Frequency domain filters operations on the image are performed on the transform image. There are many technique to transform image like, Discrete wavelet transform, Fourier transform, Discrete cosine transform. Usually everybody uses Fourier transform.

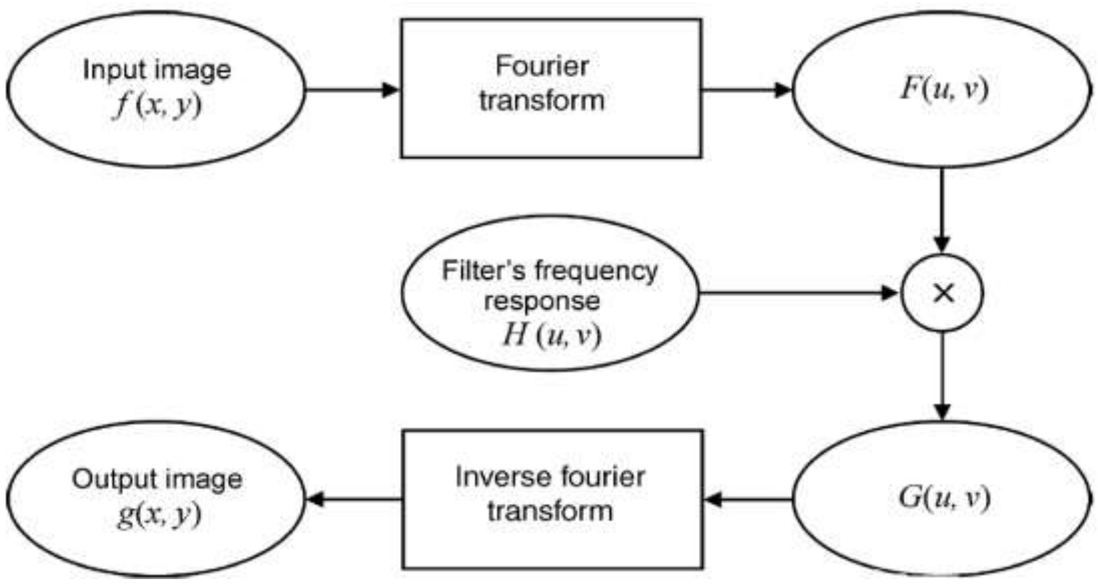


Fig 12: Frequency Domain flow chart

Input image is transformed by used Fourier transform as an output we will get Fourier transformed image and based on the image and type of filter we will get H(u,v) and multiple of both F(u,v) and H(u,v) we will get Fourier transformed output of an image on applying inverse Fourier transform we will the output image.

Differences between spatial and frequency domain:

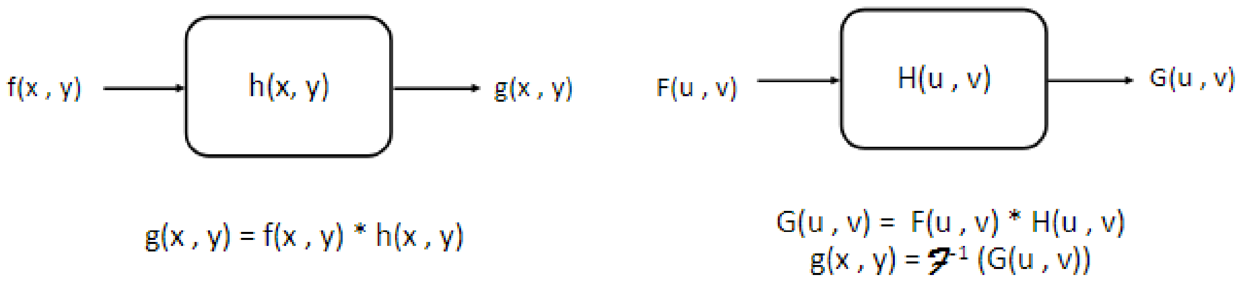

f(x , y)
h(x, y)
g(x , y)
g(x , y) = f(x , y) * h(x , y)
F(u , v)
H(u , v)
G(u , v)
G(u , v) = F(u , v) * H(u , v)
g(x , y) = 𝓕⁻¹ (G(u , v))

Fig 13: Spatial domain (b) Frequency domain

In frequency domain Fourier transformed image is multiplied to filter which is generated based on the image whereas in spatial domain, filter is convoluted directly with the input image. In frequency domain after getting the Fourier transformed output on multiplying with filter and Fourier transformed image inverse Fourier transform should be applied in order to output image but in spatial domain no operation is required after image is convoluted with kernel.

Representation

f(x,y) is an input image	F(u,v) is a Fourier transformed input
h(x,y) is a kernel	H(u,v) is filter for input image
g(x,y) is an output image	G(u,v) is a Fourier transformed output

In my project I used Fast Fourier Transform rather than using Discrete Fourier Transform because time complexity of Discrete FT O(n^2) and time complexity Fast FT is O(n log(n)).

Steps to followed are:

1. Applying 2D Fast Fourier Transform to the image by using fft2() function.

2. Applying fftshift() to the output image of 1st step in order to bring the high intensity pixels to the center of image to make spectrum more visible to human eye.

3. Applying high/low pass filter to Fourier transformed and shifted image.

4. Decentralizing the image to set the pixels in their original positions.

5. Finally apply inverse Fast Fourier Transformation to generate final output image.

Frequency domain is also divided into two types

- Smoothing filters
- Sharpening filters

Smoothing filters:

Smoothing filters are mainly used to remove noises from images. Smoothing filter are low pass filters. Low pass filters allow only low frequency to pass through. Low frequency in an image means change in pixel values should be slow. As this filter allowing only low frequencies, high frequencies like noises will fail to pass through.

Smoothing filters consists of

- Ideal low pass filter
- Butterworth low pass filter
- Gaussian low pass filter

Sharpening filters:

Sharpening filters are mainly used to highlight the edges. Sharpening filter are high pass filters. High pass filters allow only high frequency to pass through. High frequency in an image means change in pixel value should very high. The intensity change near the edges is high.so, edges are highlight in the output image.

Sharpening filters consists of

- Ideal high pass filter
- Butterworth high pass filter
- Gaussian high pass filter

Ideal pass filter:

The idea behind ideal low pass filter is the distance between pixel and center of frequency rectangle greater than threshold then H(u,v) will be Zero. If the distance is less than or equal to threshold then H(u,v) will be One

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$

Formulae for ILP

1) *Where,*

D_0 is a threshold value which should greater than Zero

D(u,v) is distance between pixel and center of frequency rectangle

Contrasts to high pass filter if the distance is greater than threshold value then H(u,v) will be One. If the distance is less than or equal to threshold value then H(u,v) will be Zero.

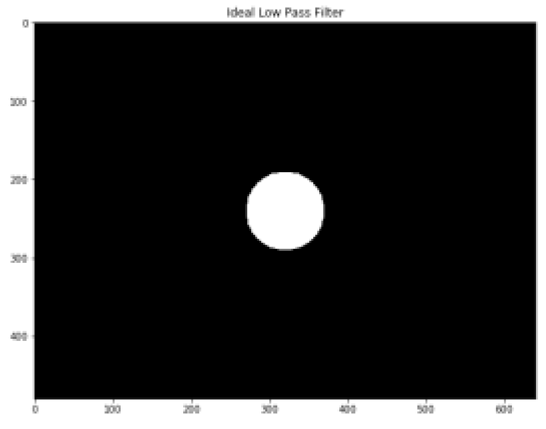
$$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \leq D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$$

Formulae for IHP

Where,

D_0 is a threshold value which should be greater than zero

D(u,v) is distance between pixel and center of frequency rectangle



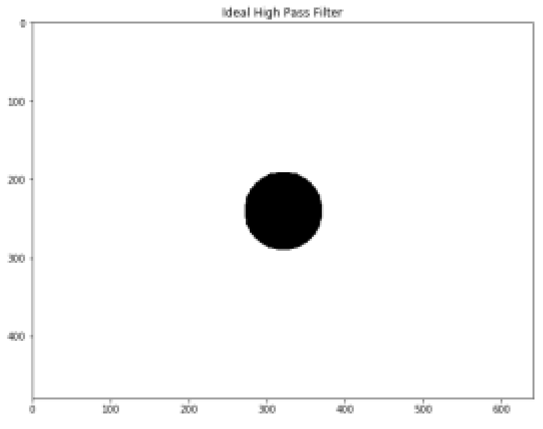


Fig 14: (a) ILP filter with D_0=50 (b) IHP filter with D_0=50

Butterworth filter:

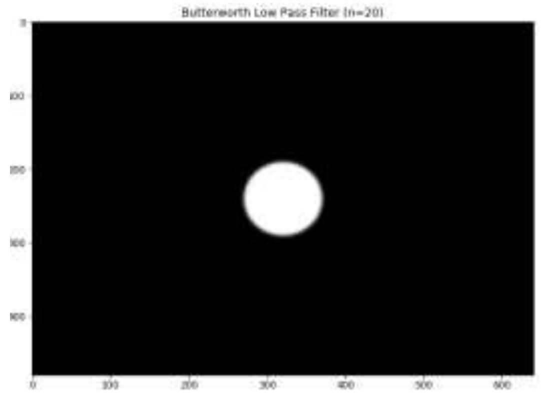
In ideal filter there is sharp discontinuity between passed and filtered frequencies. Unlike to that Butterworth filters don't have a sharp discontinuity between passed and filtered frequencies in order to achieve that Butterworth introduced a new variable 'n'. As we changing the value of n then clearness of cutoff changes. 'n' value is inversely proportional to clearness of cutoff. We can observe that clearness of cutoff for n=20 and n=3 in below figures

$$H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}} \quad H(u,v) = \frac{1}{1 + [D_0/D(u,v)]^{2n}}$$

(a) Formulae for Butterworth low pass (b) Formulae for Butterworth high pass
Where,

D_0 is threshold value should be greater than zero

D(u,v) is distance pixel and center in frequency rectangle



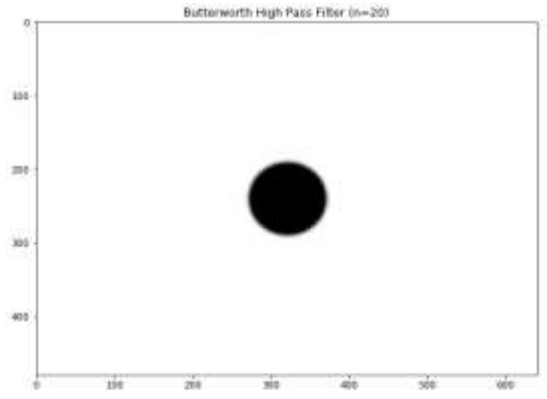
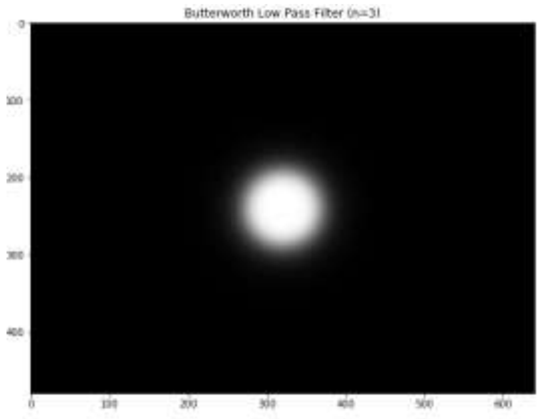


Fig 15: (a) BWLP filter with n=20 (b) BWLP filter with n=20, D_0=50



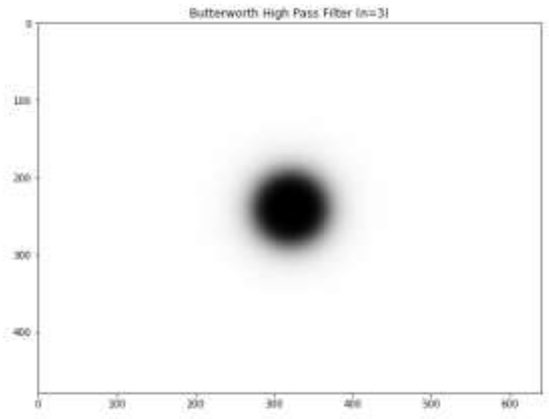


Fig 16: (c) BWLP filter with n=3, D_0=50 (d) BWLP with n=3, D_0=50

Gaussian filter:

Gaussian filter has better smoother cutoff compared to Butterworth filter. As the cutoff between filtered and passed frequencies is blur which make the look and feel of image better compared to the Ideal and Butterworth filters.

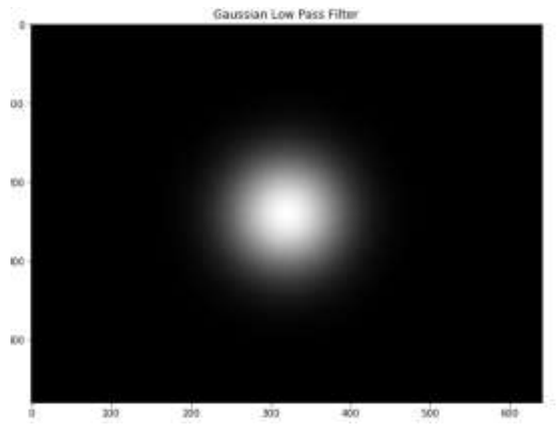
$$H(u, v) = e^{-D^2(u, v)/2D_0^2} \qquad H(u, v) = e^{-D^2(u, v)/2D_0^2}$$

(a) formulae for Gaussian low pass filter (b) formulae for Gaussian high pass filter

Where,

D_0 is threshold value should great than 0

D(u,v) is distance pixel and center in frequency rectangle



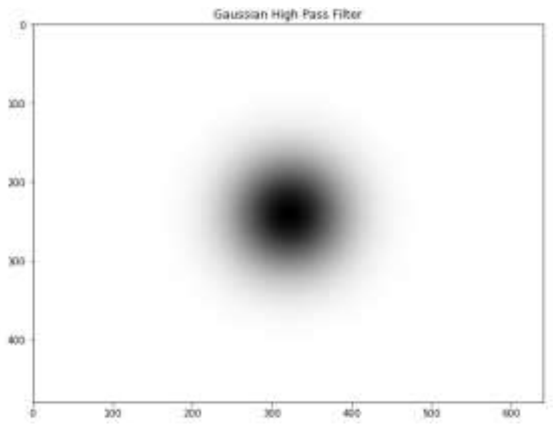


Fig 17: (a) GLP filter with D_0=50 (b) GLP filter with D_0=50

3.2 SOFTWARE REQUIREMENT SPECIFICATIONS:

A document that specifies the nature of a software model or project is commonly referred to as Software Requirement Specifiction (SRS). This can be stated as manual of project and is prepared before proceeding onto the project. We have to follow some important guidelines in preparing an efficient SRS document which can be understood easily. This consists of scope, purpose, functional and non-functional requirements of software application as well as requirements of software and hardware. This also consists of details regarding conditions of environment, safety, security, quality attributes etc.

Hardware Requirements:

Processor	: minimum core i3 processor
RAM	: minimum 2GB or 8GB maximum
Hard Disk Space	: more than 50GB

Software Requirements:

Operating System	: Windows 8 or later versions of Windows

Anaconda software

3.2.1 TECHNOLOGY DESCRIPTION:

Python:

- Python is an oop language and everyone can understand it easily.
- It is interpreted, high-level, interactive language which is used worldwide in many applications.
- The indentation feature of python improves its code readability.

Syntax and Semantics:

- Python don't have curly braces for block starting and ending instead of braces python uses indentation to represent block.
- Import statement is used to include already written modules, functions and classes.
- Class statement is used to act aas template for the object.
- Def is keyword used to create methods.
- Return statement is used to through something to the place where the function is called.
- If statement is used to execute the conditional statement and and we can even use elif or else along with that.line of if or else ends with colon.
- While is used repeat set of statement until condition is false. Usually while is used when we don't know how many to loop the statement.
- for is also used to execute a statement required number of times.
- Exceptional handling in python can be done using the statements like try and except and we can even use finally block which executes always.

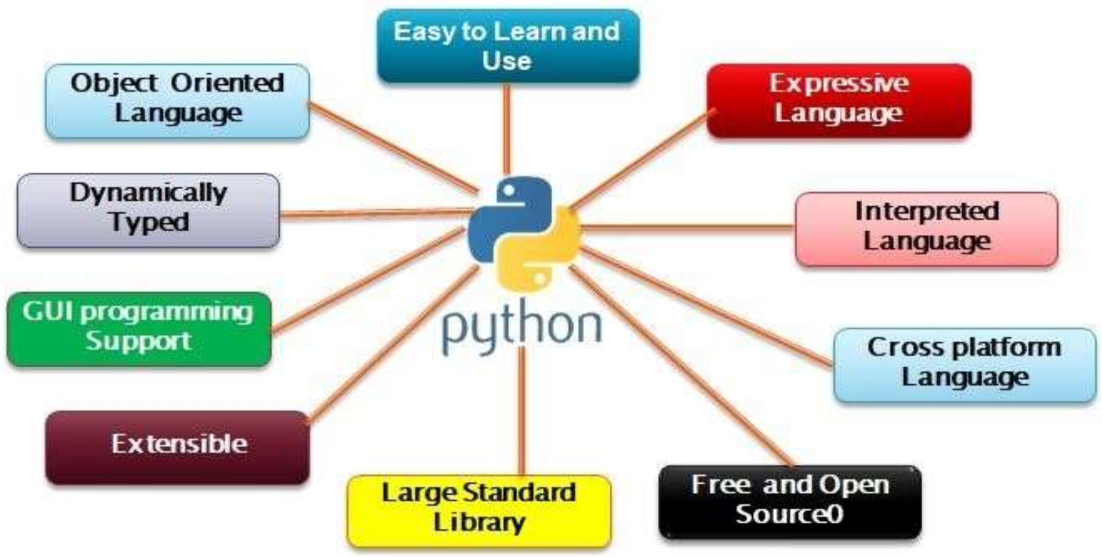


Fig 18:Feature of python

Cv2 Module:

Cv2 is a computer vision module which is used to work on images like reading image from user, cropping based on given cooridnates, Resizing an image. This module is used to convert image colors. This opencv module used to do analysis on images, advanced robots, and many applications which uses this module.

Numpy:

Numpy is derived from word Numeric python. It consists data structure with deals with matrices and multi-dimensional arrays. Using numpy we can easy handle large amount of data for suppose if we want to apply a log operation on every element we have to traverse the whole array and apply but by using numpy with function we can directly apply without traversing the array.

Tkinter:

Tkinter is popularly used GUI. This is used to provide graphical user interface to the user which help interact with our program.

Window:

Window means a rectangular box which contains all widget which help users to interact

Widget:

Widget are the components like buttons, labels,etc.

CHAPTER-4

EXPERIMENTATION AND RESULTS

4.1. Program to implement spatial and frequency domain with GUI using python

```
import cv2
import numpy as np

import matplotlib.pyplot as plt
from math import sqrt,exp
from matplotlib.backends.backend_tkagg
import (FigureCanvasTkAgg,NavigationToolbar2Tk)



import tkinter as tk

from tkinter import filedialog as fd
from tkinter import ttk

address=""
gui = tk.Tk(className='Filters')
gui.geometry("380x550")
gui.configure(bg='#000000')




def acknowledge():

    global address
    address = fd.askopenfilename()
    print(address)
    tk.messagebox.showinfo('Info','Image is uploaded')

errmsg = 'Error!'
tk.Button(text='Upload your image',font="Times 11 bold",command=acknowledge,
width=30,height=1).pack(pady = 10, padx = 100)




def display(title,*images):
    length=len(images)

    if(length>=3):
        fig = plt.figure(figsize=(20,20), constrained_layout=True)
    else:
        fig = plt.figure(figsize=(20,20), constrained_layout=False)
    # plt.xticks([]),plt.yticks([]) can be used to hide axis


    len_str=str(length)
    for i in range(1,length+1):
        temp=int('1'+len_str+str(i))
        plt.subplot(temp),plt.imshow(images[i-1],"gray"),plt.title(title[i-
1]),plt.xticks([]),plt.yticks([])


    # Toplevel object which will
    # be treated as a new window
    newWindow = tk.Toplevel(gui)

    # sets the title of the
    newWindow.title("output")

    # sets the geometry of toplevel
    if(length>3):
         # sets the title of the
        newWindow.title("Frequency Domain Filters")
        newWindow.geometry("800x200")
    else:
         # sets the title of the
        newWindow.title("Spatial Domain Filters")
        newWindow.geometry("600x300")

    newWindow.configure(bg='#ffffff')

    # creating the Tkinter canvas
    # containing the Matplotlib figure
    canvas = FigureCanvasTkAgg(fig,master = newWindow)
    canvas.draw()

    # placing the canvas on the Tkinter window
    canvas.get_tk_widget().pack()

    # creating the Matplotlib toolbar
    toolbar = NavigationToolbar2Tk(canvas,newWindow)
    toolbar.update()

    # placing the toolbar on the Tkinter window
    canvas.get_tk_widget().pack()


#====================spatial domain==========================

def convolve_linear(img,fil):
    img_h = img.shape[0]
    img_w = img.shape[1]

    H = (fil.shape[0] - 1) // 2
    W = (fil.shape[1] - 1) // 2

    out = np.zeros((img_h, img_w))
    #iterate over all the pixel of image X
    for i in range(H, img_h):
        for j in range(W, img_w):
            sum = 0
            #iterate over the filter
            for k in range(-H, H+1):
                for l in range(-W, W+1):
                    sum += (fil[H+k, W+l] * img[i+k, j+l])

            out[i,j] = sum

    return out

#----------------------linear filter-----------------------------------------
def mean():
    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    image=cv2.imread(temp,0)
    #0 incidcates read in gray scale by default
    k= np.ones((3,3),np.float32)/9
    processed_image = convolve_linear(image,k)
    title=["input image","mean filter output"]
    display(title,image,processed_image)

def gaussian():
    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    image=cv2.imread(temp,0)
    #0 incidcates read in gray scale by default
    k= np.array([[1,2,1],[2,4,2],[1,2,1]],np.float32)/16
    processed_image = convolve_linear(image,k)
    title=["input image","gaussian filter output"]
    display(title,image,processed_image)

def laplacian():
    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    image=cv2.imread(temp,0)

    k= np.array([[0,1,0],[1,-4,1],[0,1,0]])
    processed_image = convolve_linear(image,k)
    title=["input image","laplacian filter output"]
    display(title,image,processed_image)

def sobel():
    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    image=cv2.imread(temp,0)
    sob_img_x=sobel_x(image)
    sob_img_y=sobel_y(image)

    sob_out = np.sqrt(np.power(sob_img_x, 2) + np.power(sob_img_y, 2))
    # mapping values from 0 to 255
    processed_image= (sob_out / np.max(sob_out)) * 255
    title=["input image","sobel filter output"]
    display(title,image,processed_image)

def sobel_x(img):
    k= np.array([[-1,-2,-1],[0,0,0],[1,2,1]])
    sob_x = convolve_linear(img,k)
    return sob_x

def sobel_y(img):
    k= np.array([[1,2,1],[0,0,0],[-1,-2,-1]])
    sob_y = convolve_linear(img,k)
    return sob_y

#----------------------Non-Linear filters--------------------------------------
def median_3():
    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    image=cv2.imread(temp,0)

    sha = image.shape
    processed_image=image.copy()

    for i in range(1, sha[0] - 1):
        for j in range(1, sha[1] - 1):

            lst = []
            for x in range(i-1, i + 2):
                for y in range(j-1, j+2):
                    lst.append( image[x][y] )


            lst.sort()

            processed_image[i][j] = lst[4]
    title=["input image","median_3 filter output"]
    display(title,image,processed_image)


def median_5():

    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    image=cv2.imread(temp,0)


    sha = image.shape
    processed_image=image.copy()


    for i in range(1, sha[0] - 2):
        for j in range(1, sha[1] - 2):

            lst = []
            for x in range(i-2, i + 3):
                for y in range(j-2, j+3):
                    lst.append( image[x][y] )

            lst.sort()

            processed_image[i][j] = lst[12]
    title=["input image","median_5 filter output"]
    display(title,image,processed_image)

def min_filter():

    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    image=cv2.imread(temp,0)

    sha = image.shape
    processed_image=image.copy()

    for i in range(1, sha[0] - 1):
        for j in range(1, sha[1] - 1):

            lst = []
            for x in range(i-1, i + 2):
                for y in range(j-1, j+2):
                    lst.append( image[x][y] )

            lst.sort()

            processed_image[i][j] = lst[0]
    title=["input image","min filter output"]
    display(title,image,processed_image)

def max_filter():
    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    image=cv2.imread(temp,0)
    sha = image.shape
    processed_image=image.copy()


    for i in range(1, sha[0] - 1):
        for j in range(1, sha[1] - 1):

            lst = []
            for x in range(i-1, i + 2):
                for y in range(j-1, j+2):
                    lst.append( image[x][y] )
            #sort the values

            lst.sort()

            processed_image[i][j] = lst[8]
    title=["input image","max filter output"]
    display(title,image,processed_image)



#===============frequencydomain====================

#-----------------frequency domain filters calculation--------------------

def dist(p1,p2):
    return sqrt((p1[0]-p2[0])**2 + (p1[1]-p2[1])**2)

def idealLPFilter(D0,img_shape):
    filter = np.zeros(img_shape[:2])
    r, c = img_shape[:2]
    center = (r/2,c/2)
    for x in range(c):
        for y in range(r):
            if dist((y,x),center) < D0:
                filter[y,x] = 1
    return filter

def idealHPFilter(D0,img_shape):
    filter = np.ones(img_shape[:2])
    r, c = img_shape[:2]
    center = (r/2,c/2)
    for x in range(c):
        for y in range(r):
            if dist((y,x),center) < D0:
```

```
            filter[y,x] = 0
    return filter

def butterworthLPFilter(D0,img_shape,n):
    filter = np.zeros(img_shape[:2])
    r, c = img_shape[:2]
    center = (r/2,c/2)
    for x in range(c):
        for y in range(r):
            filter[y,x] = 1/(1+(dist((y,x),center)/D0)**(2*n))
    return filter

def butterworthHPFilter(D0,img_shape,n):
    filter = np.zeros(img_shape[:2])
    r, c = img_shape[:2]
    center = (r/2,c/2)
    for x in range(c):
        for y in range(r):
            filter[y,x] = 1-1/(1+(dist((y,x),center)/D0)**(2*n))
    return filter

def gaussianLPFilter(D0,img_shape):
    filter = np.zeros(img_shape[:2])
    r, c = img_shape[:2]
    center = (r/2,c/2)
    for x in range(c):
        for y in range(r):
            filter[y,x] = exp(((-dist((y,x),center)**2)/(2*(D0**2))))
    return filter

def gaussianHPFilter(D0,img_shape):
    filter = np.zeros(img_shape[:2])
    r, c = img_shape[:2]
    center = (r/2,c/2)
    for x in range(c):
        for y in range(r):
            filter[y,x] = 1 - exp(((-dist((y,x),center)**2)/(2*(D0**2))))
    return filter

#---------------------------------main---------------------------------------

def idealLowPass():
    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    img = cv2.imread(temp, 0)
    original = np.fft.fft2(img)
    center = np.fft.fftshift(original)
    LowPassCenter = center * idealFilterLP(50,img.shape)
    LowPass = np.fft.ifftshift(LowPassCenter)
    inverse_LowPass = np.fft.ifft2(LowPass)

    title=["Original Image","Spectrum","Centered Spectrum","Centered * LP
Filter","Decentralize","Ideal Low Pass"]

display(title,img,np.log(1+np.abs(original)),np.log(1+np.abs(center)),np.log(1+np.abs(Lo
wPassCenter)),np.log(1+np.abs(LowPass)),np.abs(inverse_LowPass))




def idealHighPass():

    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    img = cv2.imread(temp, 0)
    original = np.fft.fft2(img)
    center = np.fft.fftshift(original)
    HighPassCenter = center * idealFilterHP(50,img.shape)
    HighPass = np.fft.ifftshift(HighPassCenter)
    inverse_HighPass = np.fft.ifft2(HighPass)

    title=["Original Image","Spectrum","Centered Spectrum","Centered * HP
Filter","Decentralize","Ideal High Pass"]

display(title,img,np.log(1+np.abs(original)),np.log(1+np.abs(center)),np.log(1+np.abs(Hi
ghPassCenter)),np.log(1+np.abs(HighPass)),np.abs(inverse_HighPass))



def butterworthLowPass():

    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    img = cv2.imread(temp, 0)
    original = np.fft.fft2(img)
    center = np.fft.fftshift(original)
    LowPassCenter = center * butterworthLP(50,img.shape,10)
    LowPass = np.fft.ifftshift(LowPassCenter)
    inverse_LowPass = np.fft.ifft2(LowPass)

    title=["Original Image","Spectrum","Centered Spectrum","Centered * LP
Filter","Decentralize","butterworth Low Pass"]

display(title,img,np.log(1+np.abs(original)),np.log(1+np.abs(center)),np.log(1+np.abs(Lo
wPassCenter)),np.log(1+np.abs(LowPass)),np.abs(inverse_LowPass))



def butterworthHighPass():

    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    img = cv2.imread(temp, 0)
    original = np.fft.fft2(img)
    center = np.fft.fftshift(original)
    HighPassCenter = center * butterworthHP(50,img.shape,10)
    HighPass = np.fft.ifftshift(HighPassCenter)
    inverse_HighPass = np.fft.ifft2(HighPass)

    title=["Original Image","Spectrum","Centered Spectrum","Centered * HP
Filter","Decentralize","butterworth High Pass"]

display(title,img,np.log(1+np.abs(original)),np.log(1+np.abs(center)),np.log(1+np.abs(Hi
ghPassCenter)),np.log(1+np.abs(HighPass)),np.abs(inverse_HighPass))

#butterworthHighPass()

def gaussianLowPass():

    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    img = cv2.imread(temp, 0)
    original = np.fft.fft2(img)
    center = np.fft.fftshift(original)
    LowPassCenter = center * gaussianLP(50,img.shape)
    LowPass = np.fft.ifftshift(LowPassCenter)
    inverse_LowPass = np.fft.ifft2(LowPass)

    title=["Original Image","Spectrum","Centered Spectrum","Centered * LP Filter","Decentralize","Gaussian Low Pass"]

display(title,img,np.log(1+np.abs(original)),np.log(1+np.abs(center)),np.log(1+np.abs(LowPassCenter)),np.log(1+np.abs(LowPass)),np.abs(inverse_LowPass))



def gaussianHighPass():

    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    img = cv2.imread(temp, 0)
    original = np.fft.fft2(img)
    center = np.fft.fftshift(original)
    HighPassCenter = center * gaussianHP(50,img.shape)
    HighPass = np.fft.ifftshift(HighPassCenter)
    inverse_HighPass = np.fft.ifft2(HighPass)

    title=["Original Image","Spectrum","Centered Spectrum","Centered * HP Filter","Decentralize","gaussian High Pass"]

display(title,img,np.log(1+np.abs(original)),np.log(1+np.abs(center)),np.log(1+np.abs(HighPassCenter)),np.log(1+np.abs(HighPass)),np.abs(inverse_HighPass))



#-----------------------all filter at one place------------------------------

def fre_smooth_filters():
    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    img = cv2.imread(temp, 0)
    ideal_low=idealFilterLP(50,img.shape)
    butter_low = butterworthLP(50,img.shape,10)
    gaussian_low= gaussianLP(50,img.shape)
    title=["Ideal low pass","butterworth low pass","gaussian low pass"]
    display(title,ideal_low,butter_low,gaussian_low)



def fre_sharpen_filters():
    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    img = cv2.imread(temp, 0)
    ideal_high=idealFilterHP(50,img.shape)
    butter_high = butterworthHP(50,img.shape,10)
    gaussian_high= gaussianHP(50,img.shape)
    title=["Ideal high pass","butterworth high pass","gaussian high pass"]
    display(title,ideal_high,butter_high,gaussian_high)



def applied_smoothing():
    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    img = cv2.imread(temp, 0)
    original = np.fft.fft2(img)
    center = np.fft.fftshift(original)

    ideal_center=center*idealFilterLP(50,img.shape)
    butter_center = center*butterworthLP(50,img.shape,10)
    gaussian_center= center*gaussianLP(50,img.shape)

    ideal_lowpass=np.fft.ifftshift(ideal_center)
    butter_lowpass=np.fft.ifftshift(butter_center)
    gaussian_lowpass=np.fft.ifftshift(gaussian_center)

    inver_ideal_lowpass=np.fft.ifft2(ideal_lowpass)
    inver_butter_lowpass=np.fft.ifft2(butter_lowpass)
    inver_gaussian_lowpass=np.fft.ifft2(gaussian_lowpass)

    title=["Ideal low pass","butterworth low pass","gaussian low pass"]

display(title,np.abs(inver_ideal_lowpass),np.abs(inver_butter_lowpass),np.abs(inver_gau
ssian_lowpass))


def applied_sharpening():
    if(len(address)!=0):
        temp=address
    else:
        tk.messagebox.showinfo('Info','Please select Image')
        return;
    img = cv2.imread(temp, 0)
original = np.fft.fft2(img)
    center = np.fft.fftshift(original)

    ideal_center=center*idealFilterHP(50,img.shape)
    butter_center = center*butterworthHP(50,img.shape,10)
    gaussian_center= center*gaussianHP(50,img.shape)

    ideal_highpass=np.fft.ifftshift(ideal_center)
    butter_highpass=np.fft.ifftshift(butter_center)
    gaussian_highpass=np.fft.ifftshift(gaussian_center)

inver_ideal_highpass=np.fft.ifft2(ideal_highpass)
    inver_butter_highpass=np.fft.ifft2(butter_highpass)
    inver_gaussian_highpass=np.fft.ifft2(gaussian_highpass)

    title=["Ideal low pass","butterworth low pass","gaussian low pass"]

display(title,np.abs(inver_ideal_highpass),np.abs(inver_butter_highpass),np.abs(inver_ga
ussian_highpass))


#======================tkinter=========================



ttk.Separator(gui, orient='horizontal' ).pack(fill='x')
tk.Label(gui,text='Spatial domain filters',bg='black',fg='white',font = 'Times 12
italic').place(x=120,y=55)
tk.Button(text='Mean', command=mean,width=20).place(x = 25,y=90)
tk.Button(text='Gaussian', command=gaussian,width=20,height=1).place(x=200,y=90)
tk.Button(text='Laplacian', command=laplacian,width=20,height=1).place(x=25,y=135)
tk.Button(text='sobel', command=sobel,width=20,height=1).place(x=200,y=135)
tk.Button(text='median_3', command=median_3,width=20,height=1).place(x=25,y=180)
tk.Button(text='median_5',
command=median_5,width=20,height=1).place(x=200,y=180)
tk.Button(text='Min_filter', command=min_filter,width=20,height=1).place(x=25,y=225)
tk.Button(text='max_filter',
command=max_filter,width=20,height=1).place(x=200,y=225)



tk.Label(gui,text='Frequency domain filters',bg='black',fg='white',font = 'Times 12
italic').place(x=120,y=265)
tk.Button(text='Ideal Low Pass', command=idealLowPass,width=20,height=1).place(x =
25,y=305)
tk.Button(text='Ideal High Pass', command=idealHighPass,width=20,height=1).place(x=
200, y = 305)

tk.Button(text='Butterworth Low Pass',
command=butterworthLowPass,width=20,height=1).place(x = 25,y = 350)
tk.Button(text='Butterworth High Pass',
command=butterworthHighPass,width=20,height=1).place(x = 200,y = 350)


tk.Button(text='Gaussain low pass',
command=gaussianLowPass,width=20,height=1).place(x = 25,y = 395)
tk.Button(text='Gaussain High pass',
command=gaussianHighPass,width=20,height=1).place(x = 200,y = 395)

tk.Button(text='frequency low pass filters',
command=fre_smooth_filters,width=20,height=1).place(x = 25,y = 440)
tk.Button(text='frequency high pass filters',
command=fre_sharpen_filters,width=20,height=1).place(x= 200,y = 440)

tk.Button(text='All 3 smoothening',
command=applied_smoothing,width=20,height=1).place(x = 25, y= 485)
tk.Button(text='All 3 sharenping',
command=applied_sharpening,width=20,height=1).place(x=200, y = 485)
tk.mainloop()
```

4.2. RESULTS:

All the above filters are implemented using python and built a GUI with help of Tkinter module. Taking an image as input on clicking the button and added a button for every filter.

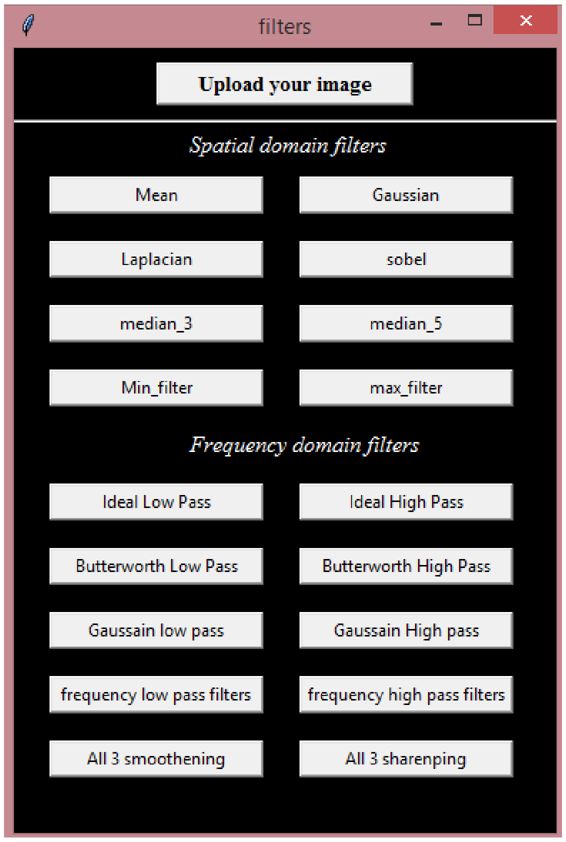
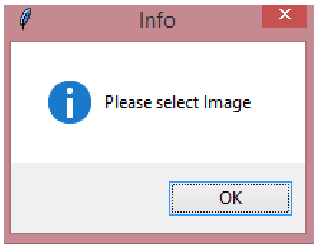


Fig 19: GUI for filters

If image is not uploaded and clicking the button of filters. It will give you acknowledgement that "please select Image" in a message box.



Please select Image

Fig 20: Acknowledgement of upload

If image is uploaded successfully then it acknowledges that "Image is uploaded".

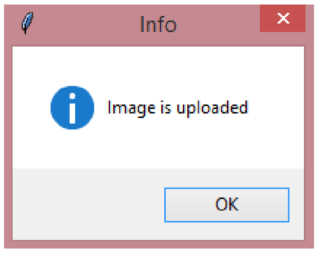


Image is uploaded

Fig 21: Acknowledgement to upload

After image is uploaded we can click any button to apply the respected filter on the selected image and display the output on the separate window. After applying a filter we can change image for next filter. They are some sample shown below.

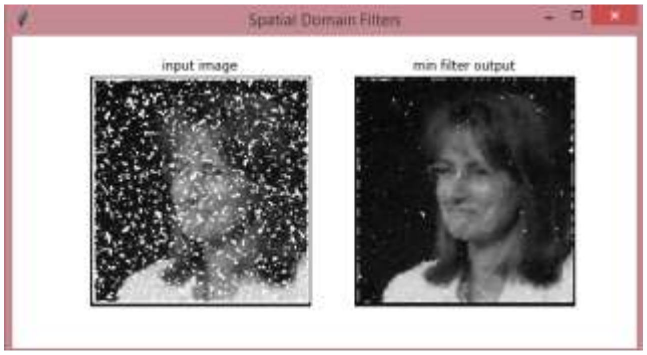


Fig 22 (a) spatial domain filter sample

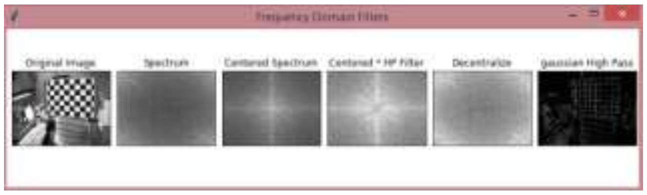


Fig 23 (b) Frequency domain filter sample

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1. Conclusions

In this paper we have discussed about types of filters which are spatial and frequency domains and spatial domain filters are falls under two categories Linear and Non-linear. Frequency domain is classified into two types' low pass and high pass. Each low and high pass contains Ideal, Butterworth, Gaussian filters.

All the above filters are implemented in python with the help of GUI. By using GUI user can apply filters on images.

5.2. Future scope:

In this project we just implemented only basic spatial and frequency domain filters with basic GUI. In future we can include lot filters in spatial as it is huge research area. We can develop a web site using python web frame works like Django, Flask and we host in a web serve so that we use that any where.We can provide a download option of an image to user which prevent that checking multiple times for same images.

REFERECES

[1] Bhisham, kushwaha, Shivam (2020), Image Filtering – Techniques, Algorithm and Applications. GIS science journal, volume 7 issue 22

[2] E.Sankar (2017), Digital image processing Image Enhancement in spatial filtering, volume 6 issue

[3] C. Narasimha and A. Nagaraja Rao (2015), A comparative study: spatial domain filter for medical image enhancement

[4] Swati Dewangan and Anup Sharma (2017), "Image Smoothening and Sharpening using Frequency Domain Filtering Technique ", IJETER Volume 5, Issue 4.

[5] Motwani, Gadiya, Rakhi C. Motwani, "Survey of Image Denoising Techniques".